LECTURE 1 : INTRODUCTION

1. Q : What is (statistical) pattern recognition?
   A : Automated recognition of patterns & regularities in data. It uses techniques developed in statistics & machine learning.

2. Q : Any differences to Machine Learning?
   A : Machine learning : learn from data without being explicitly programmed. It's almost identical to pattern recognition, since learning implies finding patterns/regularities.

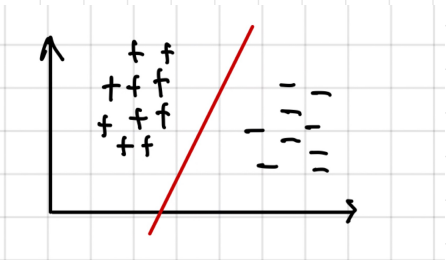
3. Q : What is machine learning? How is it different from non-learning algorithms?
   A : Non-learning algorithms are based on predetermined rules, developed by humans. The rules are fixed regardless the input data.

4. Q : What's wrong with non-learning algorithms?
   A : (1) Humans (programmers) should know the solutions & then the rules. Hence, problems humans can't solve will never be solved
   (2) Different types of data require different rules
   (3) It's human-centric & not data-centric

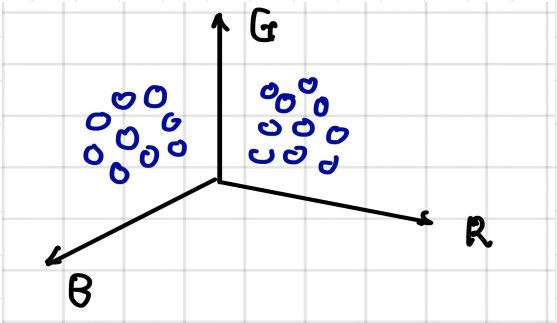
5. Typical tasks in pattern recognition :

(1) Regression

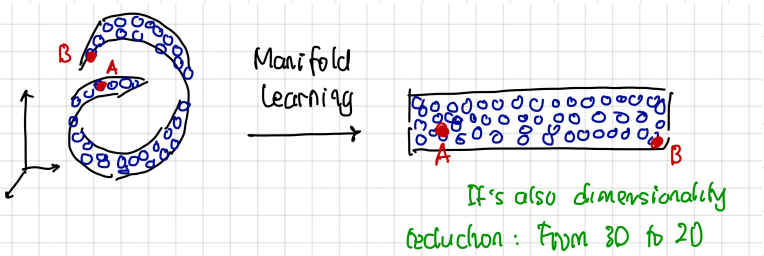
(2) Classification



(3) Clustering



(4) Dimensionality Reduction



It's also dimensionality reduction : From 3D to 2D

6. Common machine-learning framework :

Training stage

Input : - Training data
- Labels (for supervised learning)

Output : Model parameters

Goal : To learn model parameters (To find patterns)

Testing stage

Input : - Test data
- Model parameters

Output : Predicted label

Goal : To predict label of the test data.

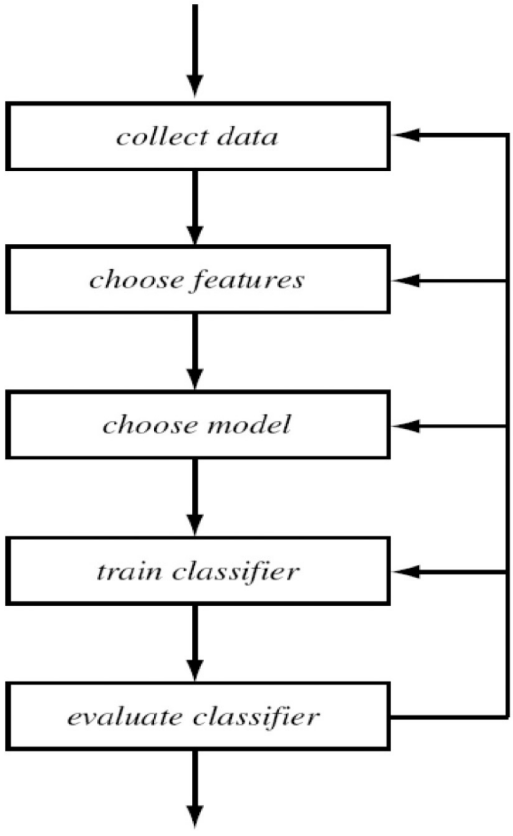
7. Types of learning (training stage) :

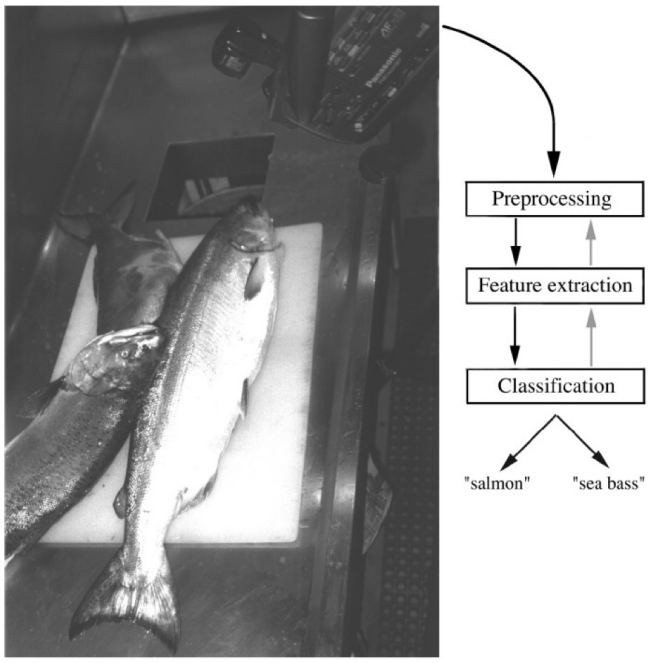
(1) Supervised : Data + Labels (= labeled data)

(2) Unsupervised : Unlabeled data

(3) Semi-supervised : Labeled data + unlabeled data

8. Design-cycle for supervised learning :

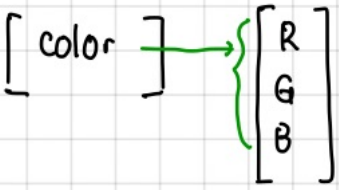




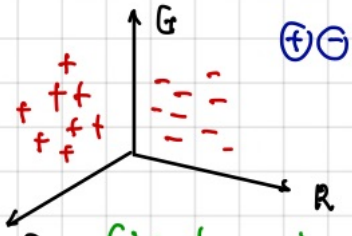
9. Example: To classify apples & oranges from their appearance:

Steps

(1) Features representation:



(2) Plot onto RGB space:

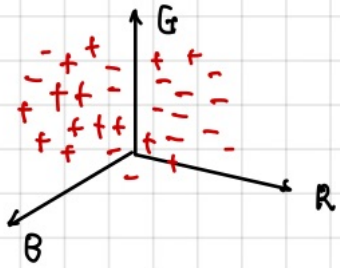


⊕⊖ : labels

(ideal case)

(3) Train an ML method to obtain a classifier (the model parameters)

Problems: color alone is insufficient:



Some apples are green/yellow

Some oranges are red/green

⇓

Adding more features: [color, edge/contour]

The dimension becomes higher (can't be observed/visualized)

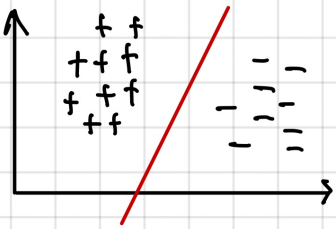
Dimensionality reduction: (to plot a high dimensionality data to a lower one: 2D/3D)

↓

Ideally: the features of different classes are separable in a higher dimension.

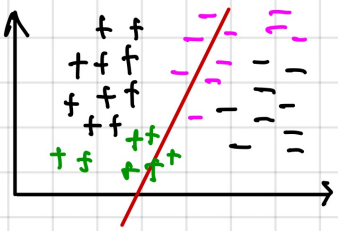
10. Overfitting problem:

Training:



Should be in the middle between the 2 classes

Testing:



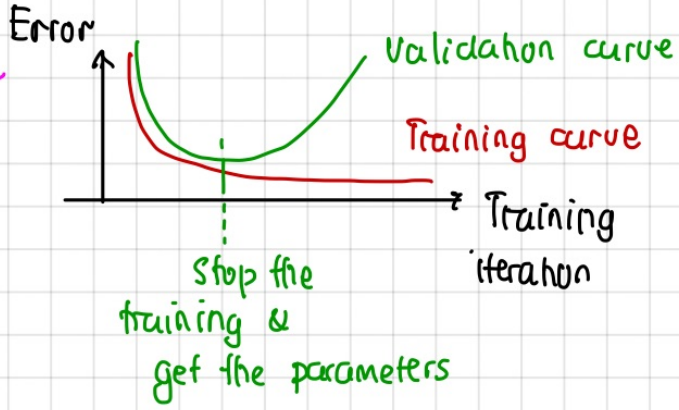
New data: test data.
the classification becomes wrong.

11. Stages in machine learning:

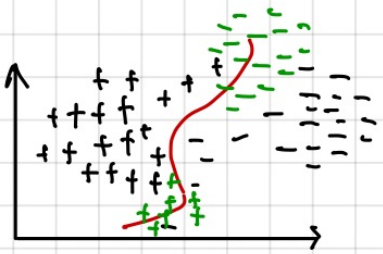
① Training
② Testing
③ Validation → to avoid overfitting

12. Validation stage:

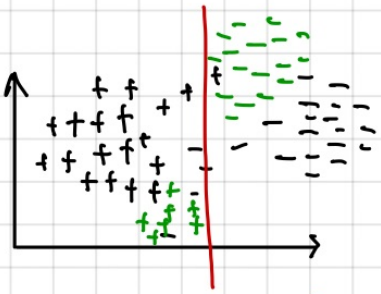
Assumption: the validation data can represent the distribution of the test data.



Example:



overfitted classifier



correct classifier

Error = classification error

13. Inductive Reasoning:

» Deductive:
1. All men are mortal
2. Socrates is a man
3. Socrates is mortal

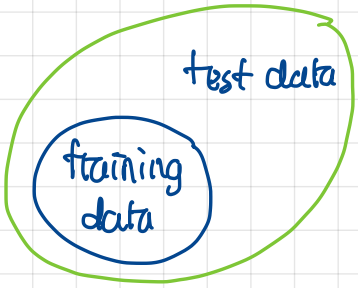
» Inductive:
1. All swans we've seen is white
2. All swans are white

↓↓

Generalization:

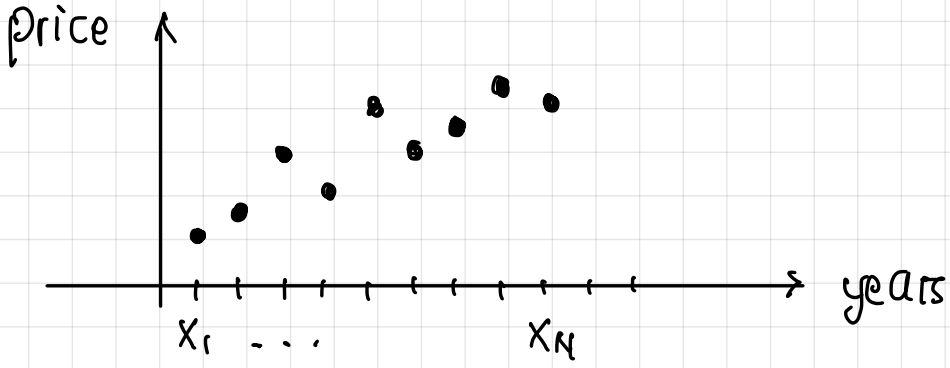
14. ML's drawbacks:

① No data, no learning
② Data dependency that can be bias
③ Inductive in nature (not always correct)



LEAST-SQUARES REGRESSION

Problem : Given a price list of a house over a period of time, how can we predict the house's price in the future ?



Solution :

(1) Data Modeling : $y(\vec{x}, w_0) = w_0$ (simplest model)

Assume : there is only one parameter to represent the whole prices.

where : $y_n = y(\vec{x}, w_0)$ = the predicted price using the model

(2) Error Function : $E(w_0) = \frac{1}{2} \sum_{n=1}^{N} (w_0 - t_n)^2$ } Least Squares

where :

t_n = the true (observed) price at time n.

N = the number of data

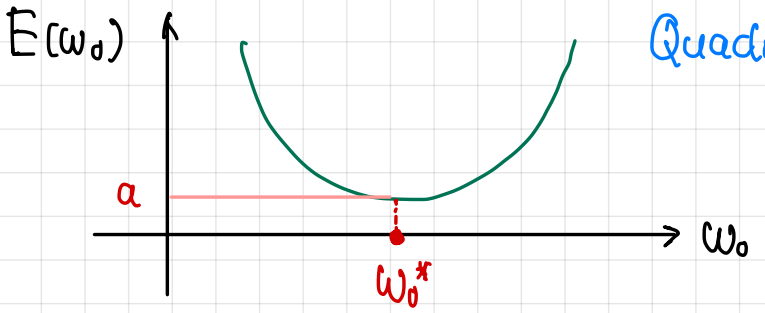
(3) Optimization (Minimization) :

$$w_0^* = \underset{\{w_0\}}{\operatorname{argmin}}\ E(w_0) = \underset{\{w_0\}}{\operatorname{argmin}}\ \frac{1}{2} \sum_{n=1}^{N} (w_0 - t_n)^2$$

w_0^* : the best esthmation of w_0

$\{w_0\}$: all the candidate values of w_0

Finding w_0^* through optimization:



Quadratic loss

$a = \min_{\{w_0\}} E(w_0)$

Q: How to get the best w_0 that minimizes the error?

A: Since the loss is quadratic:

$$\frac{\partial E(w_0)}{\partial w_0} = 0$$

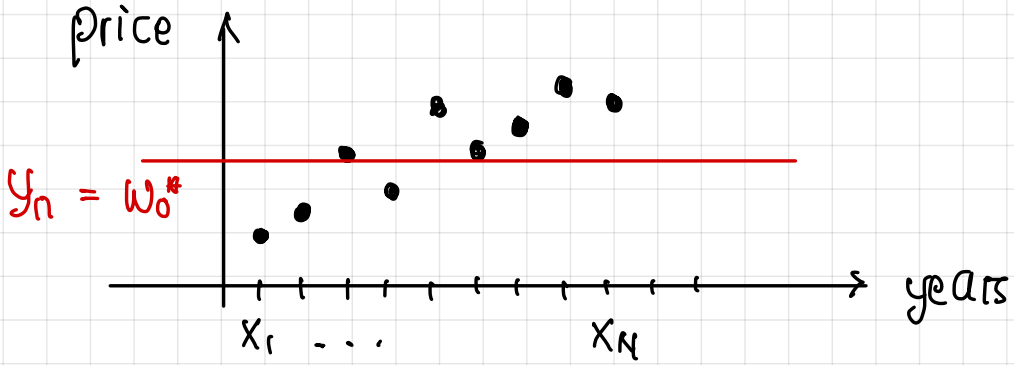
$$= \frac{1}{2} \sum_n \frac{\partial}{\partial w_0} (w_0 - t_n)^2$$

$$= \sum_n (w_0 - t_n) = 0$$

$$\sum_n w_0 = \sum t_n$$

$$w_0 = \frac{1}{N} \sum t_n$$

$\therefore$ w_0 is the mean of the prices



Better data modeling:

(1) Data modeling:

$$y(x_n, \bar{w}) = x_n w_1 + w_0 \qquad \text{(linear model)}$$

where: $\bar{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$ → model parameters

(2) Error Function:

$$E(\bar{w}) = \frac{1}{2} \sum_n \left(y(x_n, \bar{w}) - t_n \right)^2$$

(3) Optimization: $\bar{w}^* = \underset{\{\bar{w}\}}{\operatorname{argmin}} \; E(\bar{w})$

Solving the optimization:

Least squares loss is a quadratic loss, hence:

$$\frac{\partial E(\bar{w})}{\partial \bar{w}} = 0$$

meaning: $\frac{\partial E(\bar{w})}{\partial w_0} = 0$ and $\frac{\partial E(\bar{w})}{\partial w_1} = 0$

multivariable calculus

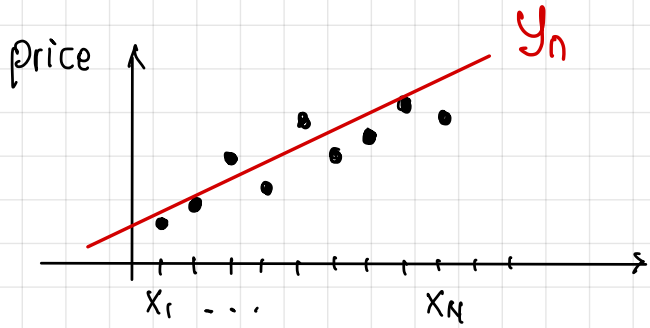
① $$\frac{\partial E(\bar{w})}{\partial w_0} = \frac{1}{2} \sum_n \left(w_0 + x_n w_1 - t_n \right)^2 = 0$$

$$= \sum_n \left(w_0 + x_n w_1 - t_n \right) = 0$$

$$w_0 = \frac{1}{N} \left(\sum_n t_n - \sum_n x_n w_1 \right)$$

② $$\frac{\partial E(\bar{w})}{\partial w_1} = \sum_n \left(w_0 + x_n w_1 - t_n \right) x_n = 0$$

$$w_1 = \left(\sum_n t_n x_n - w_0 x_n \right) \Big/ \sum_n x_n^2$$



Two equations and two unknowns (w_0 and w_1) → solvable

• General Model: Polynomial Functions

Problem: What can we do if we do not know the shape of the data distribution in advance?

(1) Data modeling:

All possible functions:

horizontal line: $y = w_0$

linear: $y = w_0 + w_1 x$

quadratic: $y = w_0 + w_1 x + w_2 x^2$

cubic: $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

⋮ ⋮

General form of polynomial functions:

$$y(x_n, \bar{w}) = \sum_{m=0}^{M-1} w_m x_n^m$$

$$\underset{M\times 1}{\bar{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}$$

M = the degree of polynomial

↓

Vector form:

$$\underset{1\times 1}{y_n} = \underset{1\times M}{\bar{w}^T} \underset{M\times 1}{\bar{x}_n}$$

;

$$y_n = y(x_n, \bar{w})$$

$$\underset{M\times 1}{\bar{x}_n} = \begin{bmatrix} x^0 \\ x^1 \\ \vdots \\ x^{M-1} \end{bmatrix}$$

Matrix form:

$$\underset{N\times 1}{\bar{y}} = \underset{N\times M}{\mathbb{X}} \underset{M\times 1}{\bar{w}}$$

$$\underset{N\times 1}{\bar{y}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} ; \quad \underset{N\times M}{\mathbb{X}} = \begin{bmatrix} x_1^0 & \cdots & x_1^{M-1} \\ \vdots & \ddots & \vdots \\ x_N^0 & \cdots & x_N^{M-1} \end{bmatrix}$$

(2) Error function :

$$E(\bar{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 \quad ; \quad y_n = y(x_n, \bar{w})$$

where :

t_n is the correct price

$y(x_n, \bar{w})$ is the estimated price : $y(x_n, \bar{w}) = \sum_{m=0}^{M} w_m x_n^m$

N = the number of the data.

In the training stage, our goal is to predict $\bar{w}$, from $\bar{x}$ (data) and $\bar{t}$ (labels).

In a vector form:

$$\underset{1\times1}{E(\bar{w})} = \frac{1}{2}\underset{N\times1}{(\bar{y}} - \underset{N\times1}{\bar{t})^T}(\bar{y} - t)$$

where : $\bar{t} = [t_1, t_2, \dots t_N]^T$

(3) Optimization :

$$\bar{w}^* = \underset{\{\bar{w}\}}{\operatorname{argmin}}\ E(\bar{w}) = \underset{\{\bar{w}\}}{\operatorname{argmin}}\ \frac{1}{2}\sum_{n=1}^{N}(\bar{w}^T\bar{x}_n - t_n)^2$$

Since the cost function is quadratic : $\underset{M\times1}{\frac{d}{d\bar{w}}E(\bar{w}) = 0}$

$$\frac{d}{d\bar{w}}\left[\frac{1}{2}\sum_{n=1}^{N}(\bar{w}^T\bar{x}_n - t_n)^2\right] = \frac{1}{2}\sum_{n=1}^{N}\frac{d}{d\bar{w}}(\bar{w}^T\bar{x}_n - t_n)^2$$

$$= \sum_{n}^{N}\underset{M\times1}{\bar{x}_n}\underset{1\times M}{(\bar{w}^T}\underset{M\times1}{\bar{x}_n} - \underset{1\times1}{t_n}) = \sum_n \bar{x}_n(\bar{w}^T\bar{x}_n) - \sum_n \bar{x}_n t_n$$

Matrix form:

$$\frac{\partial}{\partial \bar{w}} E(\bar{w}) = \underbrace{\sum_n \bar{x}_n (\bar{w}^T \bar{x}_n)}_{\left(\sum_n \bar{x}_n \bar{x}_n^T\right) \bar{w}} - \underbrace{\sum_n \bar{x}_n t_n}_{\mathbb{X}^T \bar{t}}$$

$\underbrace{\left(\sum_n \bar{x}_n \bar{x}_n^T\right)}_{M \times M}$ $\bar{w}$ ($M \times 1$)

$\mathbb{X}^T \bar{t}$ ($M \times N$ $N \times 1$)

$$= \underset{M \times N}{\mathbb{X}^T} \underset{N \times M}{\mathbb{X}} \underset{M \times 1}{\bar{w}}$$

(see the additional notes)

$$\frac{\partial}{\partial \bar{w}} E(\bar{w}) = \mathbb{X}^T \mathbb{X} \bar{w} - \mathbb{X}^T \bar{t} = 0$$

$$\mathbb{X}^T \mathbb{X} \bar{w} = \mathbb{X}^T \bar{t}$$

$$\underset{M \times 1}{\bar{w}} = \underbrace{\left(\mathbb{X}^T \mathbb{X}\right)^{-1}}_{M \times M} \underset{M \times N}{\mathbb{X}^T} \underset{N \times 1}{\bar{t}} \quad \rightarrow \quad \bar{w} = \underset{N \times M}{\mathbb{X}^\dagger} \bar{t}$$

- Inverse & pseudo inverse:

if $\mathbb{A}$ is a square & non-singular matrix, then the inverse of $\mathbb{A}$ is $\mathbb{A}^{-1}$ (non-singular: the determinant is not zero)

where $\mathbb{A}\mathbb{A}^{-1} = \mathbb{A}^{-1}\mathbb{A} = \mathbb{I}$.

But if $\mathbb{A}$ is non-square $N \times M$ matrix (where $N > M$), how to compute $\mathbb{A}^{-1}$?

Solution: pseudo-inverse: $\mathbb{A}^\dagger = (\mathbb{A}^T \mathbb{A})^{-1} \mathbb{A}^T$

Additional Notes

1. From scalar operations to vector operations):

$$\sum_{k=1}^{K} w_k x^k = \underbrace{[w_1\ w_2\ \dots\ w_M]}_{1\times M} \underbrace{\begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^M \end{bmatrix}}_{M\times 1} = w_1 x^1 + w_2 x^2 + \dots w_M x^M$$

$$= \underset{1\times M}{\bar{W}^T} \underset{M\times 1}{\bar{X}}$$

2. From Vector Operations to Matrix Operations:

$$\sum_{n=1}^{N} \underset{M\times 1}{\bar{X}_n} \underset{1\times M}{\bar{X}_n^T} = \sum_{n=1}^{N} \begin{bmatrix} x_n^1 \\ x_n^2 \\ \vdots \\ x_n^M \end{bmatrix} \begin{bmatrix} x_n^1 & x_n^2 & \dots & x_n^M \end{bmatrix}$$

$$= \sum_{n=1}^{N} \underbrace{\begin{bmatrix} x_n^1 x_n^1 & \dots & x_n^1 x_n^M \\ \vdots & \ddots & \vdots \\ x_n^M x_n^1 & \dots & x_n^M x_n^M \end{bmatrix}}_{M} \Big\}M = \begin{bmatrix} \sum_{n=1}^{N} x_n^1 x_n^1 & \dots & \sum_{n}^{N} x_n^1 x_n^M \\ \vdots & \ddots & \vdots \\ \sum_{n}^{N} x_n^M x_n^1 & \dots & \sum_{n}^{N} x_n^M x_n^M \end{bmatrix}$$

Let: $\underset{N\times M}{\mathbb{X}} = \begin{bmatrix} x_1^1 & \dots & x_1^M \\ \vdots & \ddots & \vdots \\ x_N^1 & \dots & x_N^M \end{bmatrix}$

$$\underset{M\times N}{\mathbb{X}^T} \underset{N\times M}{\mathbb{X}} = \begin{bmatrix} x_1^1 & \dots & x_N^1 \\ \vdots & \ddots & \vdots \\ x_1^M & \dots & x_N^M \end{bmatrix} \begin{bmatrix} x_1^1 & \dots & x_1^M \\ \vdots & \ddots & \vdots \\ x_N^1 & \dots & x_N^M \end{bmatrix}$$

$$= \begin{bmatrix} x_1^1 x_1^1 + x_2^1 x_2^1 + \dots + x_N^1 x_N^1 & \dots & x_1^1 x_1^M & x_2^1 x_2^M + \dots + x_N^1 x_N^M \\ \vdots & & \ddots & \vdots \\ x_1^M x_1^1 + x_2^M x_2^1 + \dots + x_N^M x_N^1 & \dots & x_1^M x_1^M + x_2^M x_2^M + \dots + x_N^M x_N^M \end{bmatrix}$$

$$= \begin{bmatrix} \sum_n^N x_n^1 x_n^1 & \dots & \sum_n^N x_n^1 x_n^M \\ \vdots & \ddots & \vdots \\ \sum_n^N x_n^M x_n^1 & \dots & \sum_n^N x_n^M x_n^M \end{bmatrix}$$

They are identical !

Recall:

$$\sum_{n=1}^{N} \underset{M\times 1}{\bar{x}_n} \underset{1\times M}{\bar{x}_n^T} = \begin{bmatrix} \sum_{n=1}^N x_n^1 x_n^1 & \dots & \sum_n^N x_n^1 x_n^M \\ \vdots & \ddots & \vdots \\ \sum_n^N x_n^M x_n^1 & \dots & \sum_n^N x_n^M x_n^M \end{bmatrix}$$

5. Matrix inverse : $\mathbb{A}\mathbb{A}^{-1} = \mathbb{A}^{-1}\mathbb{A} = \mathbb{I}$

$\mathbb{A}$ is invertible (or $\mathbb{A}^{-1}$ exists) if :

① $\mathbb{A}$ is a square matrix : $\mathbb{A}$ is a $N \times N$ matrix

② $\mathbb{A}$ is not a singular matrix (= the determinant is not zero)

6. What is a singular matrix?

A matrix whose determinant = 0 → What does it mean?

A matrix represents linear equations:

$$\left.\begin{aligned} 2x + 5y &= 1 \\ 3x + 7y &= 4 \end{aligned}\right\} \begin{bmatrix} 2 & 5 \\ 3 & 7 \end{bmatrix} \rightarrow \det \neq 0$$

↓

Thus, this matrix is invertible.
Implying the solution of x & y can be found.

$$\begin{bmatrix} 2 & 5 \\ 3 & 7 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 3 & 7 \end{bmatrix}^{-1}\begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

If a matrix is singular, we can't solve the equations:

$$\left.\begin{aligned} 2x + 5y &= 1 \\ 4x + 10y &= 2 \end{aligned}\right\} \begin{bmatrix} 2 & 5 \\ 4 & 10 \end{bmatrix} \rightarrow \det = 0$$

↳ these equations is basically the same equation. Meaning, we only have 1 equation with 2 unknowns (x & y).

7. Pseudo Inverse:

A matrix is invertible if it's square. What can we do if we have a non-square matrix: A with dimensions $N \times M$?

if $N > M$, it means we have more equations than the unknown parameters:

$$\left.\begin{aligned} 2x + 5y &= 1 \\ 3x + 4y &= 3 \\ x + 7y &= 5 \end{aligned}\right\} \begin{aligned} N &= 3 \\ M &= 2 \text{ (i.e. } x \text{ \& } y) \end{aligned}$$

↓

This is called an overdetermined system

if $N < M$, it means we have fewer equations than the unknown parameters:

$$2x + 5y = 1$$

↓

$N = 1$, $M = 2$

This is called an under determined system.

Assuming A is a $N \times M$ matrix with $N > M$, how we can get the inverse?

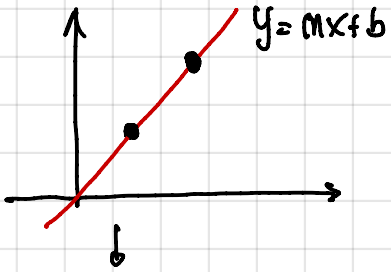
The solution is using the pseudo inverse:

$$A^{\dagger} = (A^T A)^{-1} A^T$$

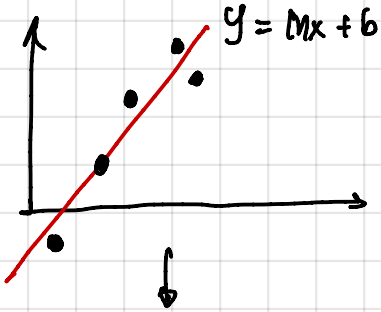
↓

However, it's not the only way to compute the pseudo inverse. There are a few alternatives, and one of them is to use SVD (Singular Value Decomposition). In python, this can be realized by calling: numpy.pinv().

8. What is the geometrical meaning of an over-determined system?



The case when we have 2 unknowns (i.e. m & b) and 2 equations (= the two dots)



The case of an over-determined system.

↓

The number of unknowns (m & b) is (much) less than the equations (5 dots)

↓

Why don't the dots lie on the line?

→ outliers or noise.

↓

outliers: when the model (in this case $y = mx + b$) is just an approximation to the data.

noise: the model is correct & precise, but the system has unexpected noise.

Questions:

Referring to our discussion in the previous pages:

1. Where is the machine learning framework (training & testing)?
2. Why come the simple least squares is called machine learning?
3. Where is the underfitting problem?
4. What does it mean by 'generalization' in machine learning?

Additional Notes:

1. Derivative:

$$u = (f(x))^n \rightarrow \frac{du}{dx} = n f(x)^{n-1} \frac{df(x)}{dx}$$

example: $u = (5x+6)^2 \rightarrow \frac{du}{dx} = 2(5x+6)5 = 10(5x+6)$

2. Partial derivative:

$$u = (f(x,y))^n \rightarrow \frac{\partial u}{\partial x} = n f(x,y)^{n-1} \frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial u}{\partial y} = n f(x,y)^{n-1} \frac{\partial f(x,y)}{\partial y}$$

example:

$$u = (5x+6y)^3 \rightarrow \frac{\partial u}{\partial x} = 3(5x+6y)^2 5 = 15(5x+6y)^2$$